Annals of Telecommunications (2020) 75:471–486
https://doi.org/10.1007/s12243-020-00767-2

Scalable kernel convex hull online support vector machine for intelligent network traffic classification

Xiaoqing Gu[1] · Tongguang Ni[1] · Yiqing Fan[2] · Weibo Wang[3]

Received: 10 June 2019 / Accepted: 22 April 2020 / Published online: 18 June 2020

Abstract
Online support vector machine (SVM) is an effective learning method in real-time network traffic classification tasks. However, due to its geometric characteristics, the traditional online SVMs are sensitive to noise and class imbalance. In this paper, a scalable kernel convex hull online SVM called SKCHO-SVM is proposed to solve this problem. SKCHO-SVM involves two stages: (1) offline leaning stage, in which the noise points are deleted and initial pin-SVM classifier is built; (2) online updating stage, in which the classifier is updated with newly arrived data points, while carrying out the classification task. The noise deleting strategy and pinball loss function ensure SKCHO-SVM insensitive to noise data flows. Based on the scalable kernel convex hull, a small amount of convex hull vertices are dynamically selected as the training data points in each class, and the obtained scalable kernel convex hull can relieve class imbalance. Theoretical analysis and numerical experiments show that SKCHO-SVM has the distinctive ability of training time and classification performance.

Keywords Online learning · Support vector machine · Scalable kernel convex hull · Network traffic classification

1 Introduction

According to the 42nd National Internet development statistics report issued by China Internet Information Center, up to December 2018 the number of Internet users in China had reached 802 million [1]. With the increase of popularity of Internet, the scale of the Internet is becoming larger and larger; meanwhile, various new network applications and services are emerging. Network technology represented by wireless communications and networks has become one of the necessities in people's daily life. Network traffic classification and application identification are the basis of solving many network management. They are of great significance to network security, intrusion detection and service quality guarantee. In the last two decades, several network traffic classification methods have been developed. The earliest traffic classifiers are mostly port-based classification methods that use the port number to identify applications [2]. The advantage of port-based classification methods is simple and fast. However, with more and more applications using dynamic port number, encryption, and re-encapsulation technology, this type of methods are no longer effective. For example, P2P applications evade identification by using port masquerading. Thus, the accuracy of port-based classification methods is less than 70% in some applications [3].

To overcome these limitations, deep packet inspection (DPI) [4] has emerged based on payload-based classification technique. DPI searches and matches the application layer load of the traffic stream, so that it can identify different types of network traffic. This type of methods can accurately identify the unencrypted traffic, but their matching process costs a lot of computation and memory. In addition, some privacy policies do not allow users to use sensitive information or check payload. The famous open source application in DPI is NDPI tool [5], which can handle encrypted traffic with a

✉ Tongguang Ni
hbxtntg-12@163.com

Xiaoqing Gu
czxqgu@163.com

Yiqing Fan
yiqingfa@usc.edu

Weibo Wang
kevinwang@outlook.com

[1] School of Information Science and Engineering, Changzhou University, Changzhou 213164, China

[2] Viterbi School of Engineering, University of Southern California, Los Angeles 90089, USA

[3] School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China

mount number of protocols. But NDPI cannot identify certain types of network traffic and labels them as unknown. The behavioral classification methods based on host behavior features can discover new network traffic behaviors without analyzing the load of the message. The research in [6] shows that the graphical connections generated from server-client models are different from the graphs from P2P applications. For instance, this type of methods carries out classification tasks by calculating the number of communication hosts and considering the number of ports and transport layer protocols. The behavioral classification methods usually have high classification accuracy with lower computational cost; however, they only work well on the end-hosts and endpoint. In addition, different network environments will invalidate the network traffic classification method based on host behavior.

With the continuous improvement of hardware and software technology, intelligent traffic classification methods based on machine learning have been widely concerned in recent years [7–12]. Machine learning methods generally do not depend on the information of application layer load, but only need the header information of packets. They establish machine learning model to achieve network traffic classification based on the statistical characteristics of network traffic. The famous example of this type of methods is intrusion detection systems, which apply machine learning methods such as neural work, decision tree, support vector machine (SVM), Bayesian network, etc. SVM becomes one of the most promising methods because SVM can solve high-dimensional nonlinear problem and maximize the margin in the kernel space [13, 14]. For example, a SVM-based method classifies the Internet traffic based on the statistical characterization of payload [15]. An incremental SVM with attenuation factor proposed in [8]. In its updating process, the support vectors and their corresponding weight will be adjusted to adopt the change of the classification hyperplane. A proximal SVM is proposed to build an online network traffic classifier, and its advantage is fast and simple system of linear equations [16].

The traditional machine learning traffic classifier trains the network traffic data as a whole for batch learning. Batch learning is suitable for small-scale classification. However, its computation complexity is high for large-scale network data. In addition, when the data distribution of network traffic changes with the time going, especially the new training data occurring, the classification performance of batch learning will degrade significantly. Thus, batch learning methods cannot adapt to practical applications of massive and streaming network traffic data. Online learning is one of the effective ways to solve this problem. Compared with traditional batch learning methods, online learning methods can achieve better generalization performance. Online learning continuously updates a classifier with continues newly arrived data points, so that its classification performance can be improved step by step. Up to now, several successful SVM-based online learning methods are proposed to handle with network traffic classification. These methods can basically be grouped into two types: (1) reducing the computation complexity by stochastic gradient descent strategy [17, 18]; (2) selecting the representative data points to keep the size of training data in a small scale [19, 20].

However, online SVMs are almost always sensitive to noise; meanwhile, they usually assume that the training data are class balanced. In practice, noise and class imbalance are common in network traffic. For example, HTTP always generates more traffic than other applications, such as P2P and VoIP [21]. In order to establish a more effective online SVM for network traffic classification, in this study, we propose a scalable kernel convex hull online SVM called SKCHO-SVM, which focuses on controlling the size of training data points based on the scalable kernel convex hull in the noise and imbalanced network traffic scenario. The SKCHO-SVM method consists of two stages: (1) discard the noise points and select the training data points based on the scalable kernel convex hull in the offline stage, then train a robust classifier on the basis of pinball-SVM; (2) re-compute the scalable kernel convex hull and update the classifier in the online stage step by step. Note that the scalable convex hull is computed in both offline and online stages. We perform this procedure periodically to control the size of training points within the reasonable scope.

The advantages of SKCHO-SVM are as follows: (1) it can deal with noise network traffic data efficiently since the outliers are discarded before the classifier training and the noise insensitivity characteristic of pinball-SVM is inherited. (2) By dynamically adjusting the scalable kernel convex hull as training data, SKCHO-SVM can be applied in the large scale network traffic scenarios. (3) The scalable kernel convex hull can relieve the class imbalance, since it exacts the convex hull vertices by the profile of the data, rather than the number of data points. (4) It is proved theoretically that the scalable kernel convex hull as training data can achieve almost the same classification performance as the classifiers trained with all the data.

The rest of this paper is organized as follows. Section 2 introduces the concept of convex hull and pinball-SVM. Section 3 describes the proposed SKCHO-SVM method in detail. Section 4 presents the classification performance analysis of SKCHO-SVM. The experimental performance of SKCHO-SVM is evaluated in Sect. 5. The conclusions are given in the final section.

2 Related works

The classification principle of traditional SVM aims to build an optimal classification hyperplane, which maximizes the minimum margin between the data points belonging to

different classes. In the binary classification task, given the dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and its corresponding class label set $\mathbf{Y} = \{y_1, y_2, \ldots, y_N\}$, the decision function of SVM classifier is.

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{1}$$

where $\mathbf{w}$ and b are the weight vector and bias parameter, respectively. Based on the maximum margin strategy, the corresponding classification hyperplane of SVM classifier can be obtained as.

$$\max_{\|f'\|=1} \left\{ minf'(\mathbf{I}) + minf'(\mathbf{II}) \right\} \tag{2}$$

where two sets of indices $\mathbf{I}$ and $\mathbf{II}$ represent as $\mathbf{I} = \{i|\ y_i = 1\}$ and $\mathbf{II} = \{i|\ y_i = -1\}$, respectively. Function $f'(\mathbf{I})$ and $f'(\mathbf{II})$ mean $f'(\mathbf{I}) = \{y_i f(\mathbf{x}_i), i \in \mathbf{I}\}$ and $f'(\mathbf{II}) = \{y_i f(\mathbf{x}_i), i \in \mathbf{II}\}$, respectively. Equation (1) shows that traditional SVM is sensitive to noise point, especially to noises around the boundary of classification hyperplane.

To solve this problem, inspired by the application of quantile in statistics, the pinball loss function is introduced into the traditional framework of SVM [22–24]. Huang et al. [22] proposed the asymmetric least squares SVM called aLS-SVM to use the pinball loss function for the least squares SVM model. Huang et al. [23] proposed an asymmetric v-tube support vector regression model to deal with noise problem in regression tasks. Huang et al. [24] proposed the pin-SVM by replacing the pinball loss function with the hinge loss function in the L1-SVM. Pin-SVM aims to maximize the quantile distance instead of maximizing the margin between two classes.

Suppose the discrete scalar set $U = \{u_1, u_2, \ldots, u_m\}$ with $u_1 \geq u_2 \geq \ldots \geq u_m$, q-lower quantile of the set U called as $min^q\{U\}$ can be written as

$$min^q\{U\} = \{t : t \in \mathbf{R}, t \text{ is } q \text{ percent larger than } u_i\}(0 \leq q \leq 1) \tag{3}$$

The classification hyperplane of pin-SVM can be obtained as

$$\max_{\|f'\|=1} \left\{ min^q f'(\mathbf{I}) + min^q f'(\mathbf{II}) \right\}. \tag{4}$$

Substituting Eq. (1) into Eqs. (3), (4), the decision function of SVM can be represented as

$$\max_{\|\mathbf{w}\|=1,b} \left\{ min^q_{i\in\mathbf{I}} y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) + min^q_{i\in\mathbf{II}} y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \right\} \tag{5}$$

Using the $\tau/(1+\tau)$-lower quantile, the pinball loss $L_\tau(u)$ in pin-SVM is defined as

$$L_\tau(u) = \begin{cases} u, & \text{if} u \geq 0. \\ -\tau u, & \text{otherwise.} \end{cases} \tag{6}$$

where pinball parameter τ is equal to $q/(1-q)$. Substituting Eq. (6) into Eq. (5), we can obtain

$$\max_{\|\mathbf{w}\|=1,b} \begin{cases} argmin \sum_{i\in\mathbf{I}} L_\tau\left(1 - y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\right) + \\ argmin \sum_{i\in\mathbf{II}} L_\tau\left(1 - y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\right) \ \} \end{cases} \tag{7}$$

From Eq. (7), we can see that maximizing the quantile distance between two classes is insensitive to noise, especially to noise around the classification hyperplane, which improves the robustness of the classifier. Based on this strategy, let $\phi(\cdot)$ be the mapping from $\mathbf{R}^d$ to the kernel space, the optimization problem of pin-SVM is represented as

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i, \\ &s.t. y_i\left(\mathbf{w}^T\phi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \\ &\quad y_i\left(\mathbf{w}^T\phi(\mathbf{x}_i) + b\right) \leq 1 + \frac{1}{\tau}\xi_i. \end{aligned} \tag{8}$$

where N is the number of training set. C is the regularization parameter, and ξ_i is the slack variable. When $\tau = 0$, pin-SVM is equivalent to the standard L1-SVM. Thus, L1-SVM can be considered as a special case of pin-SVM. Except for insensitivity to noise, it is theoretically proved that minimizing pinball loss can be viewed as a trade-off between within-class minimization and small misclassification minimization. The solution of Eq. (8) can be converted to a quadratic programming (QP) problem. The computation complexity of pin-SVM is $O(N^3)$. Therefore, pin-SVM can only deal with the classification tasks for small-scale datasets. For large-scale datasets, even for medium-sized datasets, its computation is quite large.

3 Scalable kernel convex hull online SVM

We describe the proposed SKCHO-SVM method in detail in this section. Firstly, we introduce the scalable convex hull in the kernel space in Sect. 3.1. It is the core component of SKCHO-SVM. Based on the definition of scalable kernel convex hull, a small amount of scalable convex hull vertices are computed as training points to train a classifier. In Sect. 3.2, we present a detailed description of SKCHO-SVM. In Sect. 3.3, we provide a theoretical analysis of SKCHO-SVM.

3.1 Scalable kernel convex Hull

The solution of traditional SVM is equivalent to finding an optimal classification hyperplane, which separates the contour points of two classes at the maximum intervals.

Geometrically, convex hull can accurately describe the geometric structure of data points.

Definition 1 (convex hull) [25]. The convex hull of the given dataset $\mathbf{X}$ is the smallest convex set containing all data points in $\mathbf{X}$

$$co(\mathbf{X}) = \left\{ \sum_{i=1}^{n} \alpha_i \mathbf{x}_i | \mathbf{x}_i \in \mathbf{X} | \alpha_i \geq 0 | \sum_{i=1}^{n} \alpha_i = 1 \right\} \tag{9}$$

Any point in $\mathbf{X}$ can be written as a linear combination of convex hull vertices in $co(\mathbf{X})$

$$\mathbf{x}_i = \sum_{\mathbf{x}_t \in co(\mathbf{X})} \lambda_{i,t} \mathbf{x}_t \tag{10}$$

where $\sum_{\mathbf{x}_t \in co(\mathbf{X})} \lambda_{i,t} = 1$ and $\lambda_{i,\,t} \geq 0$.

By introducing the nonlinear kernel mapping (ϕ), the input data in kernel space are represented as $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \mapsto \phi(\mathbf{X}) = \{\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)\}$. The kernel convex hull of dataset $\phi(\mathbf{X})$ is defined as $co(\phi(\mathbf{X}))$. Any point $\phi(\mathbf{x}_i)$in $\phi(\mathbf{X})$ can be written as a linear combination of convex hull vertices in $co(\phi(\mathbf{X}))$:

$$\phi(\mathbf{x}_i) = \sum_{\phi(\mathbf{x}_t) \in co(\phi(\mathbf{X}))} \mu_{i,t} \phi(\mathbf{x}_t) \tag{11}$$

where $\sum_{\phi(\mathbf{x}_t) \in co(\phi(\mathbf{X}))} \mu_{i,t} = 1$ and $\mu_{i,\,t} \geq 0$.

In order to ensure the solvability of the kernel convex hull, it is necessary to "relax" the requirements for the kernel convex hull. For this goal, a scalable threshold ε is introduced in Eq. (11), and then we obtain the definition of scalable kernel convex hull.

Definition 2 (scalable kernel convex hull). The linear relationship between kernel convex hull vertices and $\phi(\mathbf{x}_i)$in the kernel space satisfies:

$$\max_{\mathbf{x}_i \in \mathbf{X}} \min \left\| \phi(\mathbf{x}_i) - \sum_{\phi(\mathbf{x}_t) \in co(\phi(\mathbf{X}))} \mu_{i,t} \phi(\mathbf{x}_t) \right\|^2 \leq \varepsilon \tag{12}$$

where $0 \leq \mu_{i,\,t} \leq 1$ and $\sum_{\phi(\mathbf{x}_t) \in co(\phi(\mathbf{X}))} \mu_{i,t} = 1$.

The role of εhas two aspects. First, the scalable threshold ε indicates the degree to which the approximation scalability can be tolerated. Second, different ε can adjust the size of kernel convex hull. We will discuss it in detail in the Sect. 4.3. Then, each point $\phi(\mathbf{x}_i)$in the kernel space can be written as the following linear combination of kernel convex hull vertices:

$$\phi(\mathbf{x}_i) = \sum_{\phi(\mathbf{x}_t) \in co(\phi(\mathbf{X}))} \gamma_{i,t} \phi(\mathbf{x}_t) + \boldsymbol{\delta}_i \tag{13}$$

where $\|\boldsymbol{\delta}_i\|^2 \leq \varepsilon$ and $\gamma_{i,t} = \begin{cases} \mu_{i,t}, & \text{if } \phi(\mathbf{x}_t) \in co(\phi(\mathbf{X})) \text{and} \phi(\mathbf{x}_i) \notin co(\phi(\mathbf{X})), \\ 0, & \text{otherwise.} \end{cases}$.

Since ε is a very small positive constant, each component in $\boldsymbol{\delta}_i$ is very small.

Based on the definition of Eq. (10), the geometric structure of data set $\phi(\mathbf{X})$ in the kernel space can be represented by the scalable kernel convex hull. The vertexes of scalable kernel convex hull are boundary points of $\phi(\mathbf{X})$. The classification mechanism of SVM depends on finding the exact boundary in the optimal kernel space. It is reasonable to use a small amount of vertexes of scalable kernel convex hull to train a SVM-based classifier, and SKCHO-SVM can guarantee that it is safe to delete the data points inside the scalable kernel convex hull.

3.2 Construction of SKCHO-SVM method

The proposed SKCHO-SVM in this paper consists of two stages: (1) discard the noise points and compute the scalable kernel convex hull as the initial training set to train the pinball-SVM classifier; (2) re-compute the scalable kernel convex hull and update pinball-SVM classifier step by step in the online stage.

3.2.1 Compute the scalable kernel convex Hull

The noise points also called isolated points, according to their geometric distribution, which are often far away from most of the points belonging to the normal points [26]. Most of noise points are located around the boundary of the data points. That is to say, the noise points are more likely to be recognized as the vertexes of scalable kernel convex hull. Thus, we must delete the noise points before computing the scalable kernel convex hull. Inspired of [27], we definite the noise points as follows. Given a random point $\mathbf{x}_i$, we compute the distances $d_{\ker}(\mathbf{x}_i, \mathbf{x}_j)$ between $\mathbf{x}_i$ and the other point $\mathbf{x}_j(\mathbf{x}_j \in \mathbf{X}, \mathbf{x}_j \neq \mathbf{x}_i)$ in the kernel space: $d^2_{\ker}(\mathbf{x}_i, \mathbf{x}_j) = \left\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\right\|^2$. Then, we record the occurrence frequency k of the condition in which the distance $d^2_{\ker}(\mathbf{x}, \mathbf{x}_i)$ is less than θ,

$$k = \text{count}\left(d^2_{\ker}(\mathbf{x}, \mathbf{x}_i) \leq \theta\right). \tag{14}$$

If the obtained occurrence frequency k is less than the given number $\widetilde{\alpha}$, $\mathbf{x}_i$ is identified as the noise point. It is noted that we can easily perform the deleting process in parallel mode, such that its running time is greatly reduced. After deleting a large part of noise points around the boundary the data $\mathbf{X}$, we use kernel support vector data description (KSVDD) [28] to compute the minimum enclosing ball of $\mathbf{X}$, called MEB($\phi(\mathbf{X})$), which is the

smallest hard ball contains all data points in $\phi(\mathbf{X})$. Let $\mathbf{c}$ is the center of the hypersphere for MEB($\phi(\mathbf{X})$). According to the distance $d_{\ker}(\mathbf{x}_i, \mathbf{c})$ between each point $\mathbf{x}_i$ and $\mathbf{c}$ in the kernel space, all data points are sorted in descending order of $d_{\ker}(\mathbf{x}_i, \mathbf{c})$. We set the data points on the hypersphere boundary as the initial set of scalable convex hull $co(\phi(\mathbf{X}))$. Based on the descending order, each point $\mathbf{x}_i(\mathbf{x}_i \in \mathbf{X} \boldsymbol{and} \mathbf{x}_i \notin co(\phi(\mathbf{X})))$ in a sequence is checked whether it is a scalable convex hull vertex as follows

$$\min_{\boldsymbol{\mu}} \left\| \phi(\mathbf{x}_i) - \sum_{j=1}^{|co(\phi(\mathbf{X}))|} \mu_{i,j} \phi(\mathbf{x}_j) \right\|^2, \quad (15)$$
$$s.t. \phi(\mathbf{x}_i) \in co(\phi(\mathbf{X})), 0 \le \mu_{i,j} \le 1, \sum_{j=1}^{|co(\phi(\mathbf{X}))|} \mu_{i,j} = 1.$$

In this study, we consider the commonly used Gaussian kernel, $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_i)$ becomes a constant. Its value has no influence on the solution of Eq. (15). We discard this item, and then Eq. (15) can be expressed as a matrix form

$$\min_{\boldsymbol{\mu}} \boldsymbol{\mu}^T co(\phi(\mathbf{X}))^T co(\phi(\mathbf{X})) \boldsymbol{\mu} - 2\phi(\mathbf{x}_i)^T co(\phi(\mathbf{X})) \boldsymbol{\mu}, \quad (16)$$
$$s.t. \sum_{j=1}^{|co(\phi(\mathbf{X}))|} \mu_{i,j} = 1, 0 \le \mu_{i,j} \le 1.$$

Obviously, Eq. (16) can be formulated as a convex quadratic programming (QP) problem. We substitute $\boldsymbol{\mu}$ into Eq. (16), for given a threshold ε, if point $\mathbf{x}_i$ satisfies $\left\| \phi(\mathbf{x}_i) - \sum_{j=1}^{|co(\phi(\mathbf{X}))|} \mu_{i,j} \phi(\mathbf{x}_j) \right\|^2 > \varepsilon$, $\mathbf{x}_i$ will be regarded as a scalable convex hull vertex in the kernel space. Then, we add it to set $co(\phi(\mathbf{X}))$. Otherwise, the point $\mathbf{x}_i$ will be regarded as a non-scalable kernel convex hull vertex, since it can be represented linearly by the current convex hull vertexes.

3.2.2 Train pinball-SVM classifier

For a small part of noise points distributed in the region of the data, SKCHO-SVM uses pin-SVM as the classification model to further reduce the impact of potential noise points. Inherited its advantage of insensitive to noise located around the classification hyperplane, SKCHO-SVM obtains the noise insensitive classifier model by maximizing the quantile distance between two classes in the kernel space. The unconstrained original problem of SKCHO-SVM is described as

$$\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{N} \sum_{t=1}^{M} l(\mathbf{w}, b, \phi(\mathbf{x}_t)) \quad (17)$$

where M is the size of scalable kernel convex hull. $l(\mathbf{w}, b, \phi(\mathbf{x}_t))$ is pinball loss function, $l(\mathbf{w}, b, \phi(\mathbf{x}_t)) = max\{-\tau[1 - y_t(\mathbf{w}^T\phi(\mathbf{x}_t) + b)], 1 - y_t(\mathbf{w}^T\phi(\mathbf{x}_t) + b)\}$.

3.2.3 Update classifier online

In this stage, we re-compute the scalable kernel convex hull with the newly coming data points, and then update the classifier with the newly scalable convex hull. For traditional online learning methods, two strategies are often used to update the training set: one is to directly add new points to the current training set; the other is to combine the new points with the current support vectors to form a new training set. However, with the increase of training data, the training efficiency of the first strategy will have a sharp decline. It may not be suitable for real-time classification in large scale network traffic scenarios. The second strategy adopts the support vectors of the online classifier as training data, but the support vectors are only related to the classification hyperplane constructed by the classifier, and cannot comprehensively represent the contour information of the data in the kernel space. When the distribution of newly arrived points is different from that of historical training points, the classification performance of this strategy is not efficient. At this stage, we first run the noise deleting method using Eq. (14) to judge whether the newly arrived points are noise or not. Then, we use Eq. (3) to classify each newly arrived (or one batch) point $\widetilde{\mathbf{x}}$ to obtain its class label $\widetilde{y}$. If the value of its decision function is less than a limit value, we consider $\widetilde{\mathbf{x}}$ is a scalable convex hull vertex, then we add $\widetilde{\mathbf{x}}$ into the current training set

$$\left| \mathbf{w}^T \phi(\widetilde{\mathbf{x}}) + b \right| \le 1 + \beta \quad (18)$$

where β is a constant. Finally, the current pin-SVM will be re-trained by the new training set. Otherwise, both the training set and pin-SVM will not be updated. It is noted that with the increase of newly arrived points, when the number of training set is larger than a given threshold, we will re-compute the scalable kernel convex hull again using Eq. (16).

3.2.4 Algorithm description and computation complexity analysis

Based on the analysis above, the SKCHO-SVM procedure can be summarized by algorithm 1.

Algorithm 1: SKCHO-SVM

Step 1.	Delete the noise points using (14) in the positive class $\mathbf{X}^+$ and negative class $\mathbf{X}^-$, respectively;
Step 2.	Use kernel SVDD to compute the center $\mathbf{c}$ in $\mathbf{X}^+$ and $\mathbf{X}^-$, respectively;
Step 3.	Rank the data points in descending order according to the distance from each point to $\mathbf{c}$;
Step 4.	Initialize the scalable kernel convex hull $con(\phi(\mathbf{X}^+))$ and $con(\phi(\mathbf{X}^-))$ using (15) in $\mathbf{X}^+$ and $\mathbf{X}^-$, respectively;
Step 5.	According to the descending order, use (16) to judge whether each point in $\mathbf{X}^+$ and $\mathbf{X}^-$ is a convex hull vertex. If yes, add it into the scalable kernel convex hull;
Step 6.	Use the scalable kernel convex hull in $\mathbf{X}^+$ and $\mathbf{X}^-$ to construct the training set;
Step 7.	Train pinball-SVM classifier using the training set obtained in Step 6;
Step 8.	When a new point $\mathbf{x}$ arrives, check it whether a noise point. If yes, delete it; otherwise, go to Step 9;
Step 9.	If the new point $\mathbf{x}$ satisfies (18), add $\mathbf{x}$ into the current training set and than update the Pinball-SVM classifier;
Step 10.	If the size of the current training set is large than α, go to Step 2; otherwise, go to Step 8.

Here, we discuss the computation complexity of SKCHO-SVM. In the first stage of SKCHO-SVM, we use the sequential minimum optimization (SMO) technology to solve Eqs. (15), (16) in a progressive way, and the scalable convex hull of $\mathbf{X}$ is obtained progressively. The time complexity of first stage is $O(M_0^2)$, where M_0 is the size of boundary points of KSVDD. In the second stage of SKCHO-SVM, we also use SMO technology to train pinball-SVM, and its time complexity is $O(M)$, where M is the size of scalable convex hull obtained in the first stage. The third stage is online update sage of SKCHO-SVM. We use Eq. (18) to judge the newly arrived point whether it is a scalable convex hull vertex. The computation complexity of this stage is linear. Thus, the total computation complexity of SKCHO-SVM is $O(M_0^2 + M^2)$. The value of M is much smaller than the training point size, so that SKCHO-SVM can be applied to large-scale online classification tasks. In addition, the QP solution of SKCHO-SVM can guarantee its global optimal solution.

4 SKCHO-SVM performance analysis

To theoretically analyze the classification performance of scalable kernel convex hull, we substitute all training points into the unconstrained objective function of Pinball-SVM and name it as $F_1(\mathbf{w}, b)$:

$$\min_{\mathbf{w},b} F_1(\mathbf{w},b) = \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N} l(\mathbf{w},b,\phi(\mathbf{x}_i)) \tag{19}$$

where $l(\mathbf{w}, b, \phi(\mathbf{x}_i)) = max\ \{-\tau[1 - y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b)], 1 - y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b)\}$.

We name the unconstrained objective function of SKCHO-SVM, i.e., Eq. (17), as $F_2(\mathbf{w}, b)$. In order to better compare the relationship between $F_1(\mathbf{w}, b)$ and $F_2(\mathbf{w}, b)$, using $r_{i,t}(1 \le i \le N, 1 \le t \le M)$ obtained in Eq. (13) to construct a new unconstrained objective function, name it as $F_3(\mathbf{w}, b)$:

$$\min_{\mathbf{w},b} F_3(\mathbf{w},b) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N} l(\mathbf{w},b,\mathbf{u}_i) \tag{20}$$

where $l(\mathbf{w}, b, \mathbf{u}_i) = max\ \{-\tau[1 - y_i(\mathbf{w}^T\mathbf{u}_i + b)], 1 - y_i(\mathbf{w}^T\mathbf{u}_i + b)\}$, $\mathbf{u}_i = \sum_{t=1}^{M} r_{i,t}\phi(\mathbf{x}_t), \phi(\mathbf{x}_t) \in con(\phi(\mathbf{X}))$. Different from Eq. (19), N training points in (20) is linearly represented by scalable kernel convex hull.

Theorem 1. $F_3(\mathrm{w}, b)$ and $F_2(\mathbf{w}, b)$ are defined in Eqs. (20) and (17), respectively. $F_3(\mathbf{w}, b) \le F_2(\mathbf{w}, b)$. Proof. Since $\sum_{t=1}^{M} r_{i,t} = 1$, we have the following:

$$\begin{aligned}
L_3(w,b,Z^*) &= \frac{C}{N}\sum_{i=1}^{N} max\left\{-\tau\left[1-y_i\left(w^T\sum_{t=1}^{M} r_{i,t}\phi(x_t)+b\right)\right],\ 1-y_i\left(w^T\sum_{t=1}^{M} r_{i,t}\phi(x_t)+b\right)\right\} \\
&= \frac{C}{N}\sum_{i=1}^{N} max\left\{-\tau\sum_{t=1}^{M} r_{i,t}\left[1-y_i\left(w^T\phi(x_t)+b\right)\right],\ \sum_{t=1}^{M} r_{i,t}\left[1-y_i\left(w^T\phi(x_t)+b\right)\right]\right\} \\
&\le \frac{C}{N}\sum_{i=1}^{N}\sum_{t=1}^{M} max\left\{-\tau r_{i,t}\left[1-y_i\left(w^T\phi(x_t)+b\right)\right],\ r_{i,t}\left[1-y_i\left(w^T\phi(x_t)+b\right)\right]\right\} \\
&= \frac{C}{N}\sum_{t=1}^{M} max\left\{-\tau\left[1-y_i\left(w^T\phi(x_t)+b\right)\right],\ \left[1-y_i\left(w^T\phi(x_t)+b\right)\right]\right\}\sum_{i=1}^{N} r_{i,t} \\
&= L_2(w,b,\phi(X))
\end{aligned} \tag{21}$$

Adding the term $(1/2)\|\mathbf{w}\|^2$ on the both side of above equation, we have $F_3(\mathbf{w}, b) \le F_2(\mathbf{w}, b)$. ■.

Theorem 2. $-\frac{C}{N}\sum_{i=1}^{N} max\{y_i\mathbf{w}^T\boldsymbol{\delta}_i, -\tau y_i\mathbf{w}^T\boldsymbol{\delta}_i\} \le F_1(\mathbf{w},b) - F_3(\mathbf{w},b) \le \frac{C}{N}\sum_{i=1}^{N} max\{-y_i\mathbf{w}^T\boldsymbol{\delta}_i, \tau y_i\mathbf{w}^T\boldsymbol{\delta}_i\}$

Proof.

$$
\begin{aligned}
L_1(w,b,X) &= \frac{C}{N}\sum_{i=1}^{N} max\{-\tau\left[1-y_i\left(w^T\phi(x_i)+b\right)\right],\ 1-y_i\left(w^T\phi(x_i)+b\right)\} \\
&= \frac{C}{N}\sum_{i=1}^{N} max\left\{-\tau\left[1-y_i\left(w^T\left(\sum_{t=1}^{M} r_{i,t}\phi(x_t)+\delta_i\right)+b\right)\right],\ 1-y_i\left(w^T\left(\sum_{t=1}^{M} r_{i,t}\phi(x_t)+\delta_i\right)+b\right)\right\} \\
&= \frac{C}{N}\sum_{i=1}^{N} max\left\{-\tau\sum_{t=1}^{M} r_{i,t}\left[1-y_i\left(w^T\phi(x_t)+b\right)\right]+\tau y_i w^T\delta_i,\ \sum_{t=1}^{M} r_{i,t}\left[1-y_i\left(w^T\phi(x_t)+b\right)\right]-y_i w^T\delta_i\right\} \\
&\le \frac{C}{N}\sum_{i=1}^{N} max\left\{-\tau\sum_{t=1}^{M} r_{i,t}\left[1-y_i\left(w^T\phi(x_t)+b\right)\right],\ \sum_{t=1}^{M} r_{i,t}\left[1-y_i\left(w^T\phi(x_t)+b\right)\right]\right\}+\frac{C}{N}\sum_{i=1}^{N} max\{-y_i w^T\delta_i, \tau y_i w^T\delta_i\} \\
&= L_3(w,b,X)+\frac{C}{N}\sum_{i=1}^{N} max\{-y_i w^T\delta_i, \tau y_i w^T\delta_i\}
\end{aligned}
\tag{22}
$$

(22)

Adding the term $(1/2)\|\mathbf{w}\|^2$ on the both side of abov equality, we have $F_1(\mathbf{w},b)\text{-}F_3(\mathbf{w},b) \le \frac{C}{N}\sum_{i=1}^{N} max\{-y_i\mathbf{w}^T\boldsymbol{\delta}_i, \tau y_i\mathbf{w}^T\boldsymbol{\delta}_i\}$. Similarly,

$$
\begin{aligned}
L_1(w,b,X) &= \frac{C}{N}\sum_{i=1}^{N} max\{-\tau\left[1-y_i\left(w^T\phi(x_i)+b\right)\right],\ 1-y_i\left(w^T\phi(x_i)+b\right)\} \\
&\ge \frac{C}{N}\sum_{i=1}^{N} max\left\{-\tau\sum_{t=1}^{M} r_{i,t}\left[1-y_i\left(w^T\phi(x_t)+b\right)\right],\ \sum_{t=1}^{M} r_{i,t}\left[1-y_i\left(w^T\phi(x_t)+b\right)\right]\right\}-\frac{C}{N}\sum_{i=1}^{N} max\{y_i w^T\delta_i, -\tau y_i w^T\delta_i\} \\
&= L_3(w,b,X)-\frac{C}{N}\sum_{i=1}^{N} max\{y_i w^T\delta_i, -\tau y_i w^T\delta_i\}
\end{aligned}
\tag{23}
$$

(23)

Thus we have $-\frac{C}{N}\sum_{i=1}^{N} max\{y_i\mathbf{w}^T\boldsymbol{\delta}_i, -\tau y_i\mathbf{w}^T\boldsymbol{\delta}_i\} \le F_1(\mathbf{w},b) - F_3(\mathbf{w},b) \le \frac{C}{N}\sum_{i=1}^{N} max\{-y_i\mathbf{w}^T\boldsymbol{\delta}_i, \tau y_i\mathbf{w}^T\boldsymbol{\delta}_i\}$. ■

Theorem 3. Let $(\mathbf{w}_1^*, b_1^*)$ is the optimal solution of $F_1(\mathbf{w}, b)$, $(\mathbf{w}_2^*, b_2^*)$ is the optimal solution of $F_2(\mathbf{w}, b)$. $F_1(\mathbf{w}_1^*, b_1^*) - F_2(\mathbf{w}_2^*, b_2^*) \le C\sqrt{C\varepsilon}$. Proof. According to the duality theorem of pinball-SVM, we can get $\|\mathbf{w}_2^*\| \le \sqrt{C}$.

From Theorem 2,

$$
\begin{aligned}
F_1(w_1^*,\ b_1^*) - F_3(w_2^*,\ b_2^*) &\le \frac{C}{N}\sum_{i=1}^{N} max\left\{-y_i w_2^{*T}\delta_i, \tau y_i w_2^{*T}\delta_i\right\} \\
&\le \frac{C}{N}\sum_{i=1}^{N}\|w_2^*\|\|\delta_i\| \le \frac{C}{N}\sum_{i=1}^{N}\sqrt{C\varepsilon} \\
&= C\sqrt{C\varepsilon}
\end{aligned}
\tag{24}
$$

$(\mathbf{w}_1^*, b_1^*)$ is the optimal solution of $F_1(\mathbf{w}, b)$, thus $F_1(\mathbf{w}_1^*, b_1^*) \le F_1(\mathbf{w}_2^*, b_2^*)$. Using Theorem 1, we can have:

$$
\begin{aligned}
F_1(w_1^*,\ b_1^*) - F_2(w_2^*,\ b_2^*) &\le F_1(w_1^*,\ b_1^*) - F_3(w_2^*,\ b_2^*) \\
&\le F_1(w_2^*,\ b_2^*) - F_3(w_2^*,\ b_2^*) \\
&= C\sqrt{C\varepsilon}.
\end{aligned}
$$

Theorem 4. $F_1(\mathbf{w}_2^*, b_2^*) - F_1(\mathbf{w}_1^*, b_1^*) \le 2C\sqrt{C\varepsilon}$. Proof. Let $(\mathbf{w}_3^*, b_3^*)$ is the optimal solution of $F_3(\mathbf{w}, b)$, $(\boldsymbol{\alpha}_3, \widehat{\boldsymbol{\alpha}}_3)$ is the optimal solution of $L_3(\boldsymbol{\alpha}_3, \widehat{\boldsymbol{\alpha}}_3)$, which is the dual form of $F_3(\mathbf{w}, b)$, thus $F_3(\mathbf{w}_3^*, b_3^*) = L_3(\boldsymbol{\alpha}_3, \widehat{\boldsymbol{\alpha}}_3)$. Similarly, $(\boldsymbol{\alpha}_2, \widehat{\boldsymbol{\alpha}}_2)$ is the optimal solution of $L_2(\boldsymbol{\alpha}_2, \widehat{\boldsymbol{\alpha}}_2)$, which is the dual form of $F_2(\mathbf{w}, b)$, thus $F_2(\mathbf{w}_2^*, b_2^*) = L_2(\boldsymbol{\alpha}_2, \widehat{\boldsymbol{\alpha}}_2)$. Suppose $h(\widetilde{\boldsymbol{\alpha}}, \overline{\boldsymbol{\alpha}}) = \left\{(a_t, \widehat{a}_t) : a_t = \sum_{i=1}^{N} r_{i,t}\widetilde{a}_i \textit{ and } \widehat{a}_t = \sum_{i=1}^{N} r_{i,t}\overline{a}_i\right\}$, then $h(\widetilde{\boldsymbol{\alpha}}_2, \overline{\boldsymbol{\alpha}}_2) = (\boldsymbol{\alpha}_2, \widehat{\boldsymbol{\alpha}}_2)$, thus $(\widetilde{\boldsymbol{\alpha}}_2, \overline{\boldsymbol{\alpha}}_2)$ is also the optimal solution of $L_3(\boldsymbol{\alpha}_3, \widehat{\boldsymbol{\alpha}}_3)$:

$$
\begin{aligned}
L_2\left(h\left(\widetilde{\alpha}_2,\ \overline{\alpha}_2\right)\right) &= \sum_{t=1}^{M}\alpha_t - \frac{1}{2}\sum_{t=1}^{M}\sum_{s=1}^{M}\alpha_t\alpha_s y_t y_s\phi\left(x_t\phi(x_s)\right. \\
&= \sum_{t=1}^{M}\sum_{i=1}^{N} r_{i,t}\widetilde{a}_i - \frac{1}{2}\sum_{t=1}^{M}\sum_{s=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N} r_{i,t}r_{j,s}\widetilde{a}_i\widetilde{a}_j y_t y_s\phi\left(x_t\phi(x_s)\right.
\end{aligned}
\tag{25}
$$

Table 1 The number of traffic flows of each class in four network traffic datasets

Traffic classes	Day1	Day2	Day3	SiteB
WWW	274,977	140,875	218,620	212,492
MAIL	28,124	16,487	3978	10,871
BULK	12,151	10,793	5351	555
ATTACK	1751	987	35	4008
CHAT	0	0	66	506
P2P	2085	2762	22,287	17,851
DATABASE	2794	2606	9181	0
MULTIMEDIA	496	4	19	11
VOIP	0	0	93	1043
SERVICES	1808	1111	70	465
INTERACTIVE	86	36	323	317
GAMES	5	0	0	150
GRID	0	1	0	93
Total	324,277	175,578	260,074	248,362

Since $\sum_{t=1}^{M} r_{i,t} = 1$, the above formula can be simplified as follows:

$$L_2\left(h\left(\widetilde{\boldsymbol{\alpha}}_2, \overline{\boldsymbol{\alpha}}_2\right)\right) = \sum_{i=1}^{N} \widetilde{a}_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \widetilde{a}_i \widetilde{a}_j y_t y_s \phi(\mathbf{x}_t)^T \phi(\mathbf{x}_s) = L_3\left(\boldsymbol{\alpha}_2, \widehat{\boldsymbol{\alpha}}_2\right) \tag{26}$$

Thus, $L_2(h(\widetilde{\boldsymbol{\alpha}}_2, \overline{\boldsymbol{\alpha}}_2)) = L_2(\boldsymbol{\alpha}_2, \widehat{\boldsymbol{\alpha}}_2) = F_2\left(\mathbf{w}_2^*, b_2^*\right) = L_3(\boldsymbol{\alpha}_2, \widehat{\boldsymbol{\alpha}}_2)$. At the same time, $F_3\left(\mathbf{w}_3^*, b_3^*\right) = L_3(\boldsymbol{\alpha}_3, \widehat{\boldsymbol{\alpha}}_3)$. We can have $L_3(\boldsymbol{\alpha}_3, \widehat{\boldsymbol{\alpha}}_3) \geq L_3(\boldsymbol{\alpha}_2, \widehat{\boldsymbol{\alpha}}_2)$, i.e.,

$$F_3\left(\mathbf{w}_3^*, b_3^*\right) \geq F_2\left(\mathbf{w}_2^*, b_2^*\right) \tag{27}$$

From Theorem 1, $F_3\left(\mathbf{w}_3^*, b_3^*\right) \leq F_3\left(\mathbf{w}_2^*, b_2^*\right) \leq F_2\left(\mathbf{w}_2^*, b_2^*\right)$. Thus,

$$F_3\left(\mathbf{w}_3^*, b_3^*\right) = F_3\left(\mathbf{w}_2^*, b_2^*\right). \tag{28}$$

We can further obtain that

$$-\frac{C}{N} \sum_{i=1}^{N} max\left\{y_i \mathbf{w}_1^{*T} \boldsymbol{\delta}_i, -\tau y_i \mathbf{w}_1^{*T} \boldsymbol{\delta}_i\right\} \leq F_1\left(\mathbf{w}_1^*, b_1^*\right) - F_3\left(\mathbf{w}_1^*, b_1^*\right) \tag{29}$$

Table 2 Average percentage of scalable kernel convex hull vertices in corresponding class (%) in noise mode 1

		$\sigma = 10^{-2}$	$\sigma = 10^{-1}$	$\sigma = 10^{0}$	$\sigma = 10^{1}$	$\sigma = 10^{2}$
WWW in Day1	$\varepsilon = 10^{-2}$	0.567	0.975	1.934	4.871	7.456
	$\varepsilon = 10^{-3}$	0.864	1.232	2.308	6.970	8.338
	$\varepsilon = 10^{-4}$	1.296	2.779	4.902	9.532	11.550
WWW in Day2	$\varepsilon = 10^{-2}$	0.571	0.980	1.941	4.799	7.498
	$\varepsilon = 10^{-3}$	0.859	1.238	2.311	6.957	8.328
	$\varepsilon = 10^{-4}$	1.292	2.768	4.900	9.535	11.547
WWW in Day3	$\varepsilon = 10^{-2}$	0.561	0.980	1.938	4.879	7.452
	$\varepsilon = 10^{-3}$	0.870	1.239	2.312	6.980	8.345
	$\varepsilon = 10^{-4}$	1.289	2.772	4.905	9.529	11.548
WWW in SiteB	$\varepsilon = 10^{-2}$	0.572	0.972	1.929	4.883	7.459
	$\varepsilon = 10^{-3}$	0.869	1.240	2.311	6.965	8.332
	$\varepsilon = 10^{-4}$	1.301	2.782	4.908	9.539	11.559
MAIL in Day1	$\varepsilon = 10^{-2}$	0.573	0.978	1.941	4.882	7.461
	$\varepsilon = 10^{-3}$	0.870	1.245	2.312	6.982	8.341
	$\varepsilon = 10^{-4}$	1.206	2.790	4.921	9.547	11.569
P2P in Day3	$\varepsilon = 10^{-2}$	0.574	0.983	1.942	4.883	7.462
	$\varepsilon = 10^{-3}$	0.883	1.252	2.324	6.994	8.358
	$\varepsilon = 10^{-4}$	1.308	2.788	4.914	9.556	11.574

Table 3 Average running time of computing scalable kernel convex hull vertices in corresponding class (seconds) in noise mode 1

		$\sigma = 10^{-2}$	$\sigma = 10^{-1}$	$\sigma = 10^{0}$	$\sigma = 10^{1}$	$\sigma = 10^{2}$
WWW in Day1	$\varepsilon = 10^{-2}$	0.78	2.45	5.29	9.50	12.32
	$\varepsilon = 10^{-3}$	1.46	5.56	7.11	12.53	14.38
	$\varepsilon = 10^{-4}$	2.29	9.73	15.83	17.02	20.55
WWW in Day2	$\varepsilon = 10^{-2}$	0.77	2.46	5.27	9.47	12.33
	$\varepsilon = 10^{-3}$	1.50	5.56	7.10	12.50	14.40
	$\varepsilon = 10^{-4}$	2.30	9.72	15.80	17.00	20.58
WWW in Day3	$\varepsilon = 10^{-2}$	0.76	2.43	5.28	9.48	12.30
	$\varepsilon = 10^{-3}$	1.45	5.54	7.12	12.54	14.39
	$\varepsilon = 10^{-4}$	2.28	9.72	15.82	17.00	20.54
WWW in SiteB	$\varepsilon = 10^{-2}$	0.80	2.48	5.19	9.55	12.47
	$\varepsilon = 10^{-3}$	1.54	5.49	7.02	13.00	14.65
	$\varepsilon = 10^{-4}$	2.27	9.69	15.80	17.14	20.75
MAIL in Day1	$\varepsilon = 10^{-2}$	0.06	0.26	0.63	1.01	1.30
	$\varepsilon = 10^{-3}$	0.09	0.67	0.89	1.39	1.42
	$\varepsilon = 10^{-4}$	0.19	0.88	1.42	1.67	2.12
P2P in Day3	$\varepsilon = 10^{-2}$	0.04	0.18	0.42	0.84	1.08
	$\varepsilon = 10^{-3}$	0.08	0.44	0.79	1.20	1.49
	$\varepsilon = 10^{-4}$	0.14	0.70	1.25	1.53	1.67

$$F_1(\mathbf{w}_2^*, b_2^*) - F_3(\mathbf{w}_2^*, b_2^*) \le \frac{C}{N} \sum_{i=1}^{N} max\left\{-y_i \mathbf{w}_2^{*T} \delta_i, \tau y_i \mathbf{w}_2^{*T} \delta_i\right\} \quad (30)$$

Thus,

$$\le \frac{C}{N} \sum_{i=1}^{N} \left[max\left\{-y_i \mathbf{w}_2^{*T} \delta_i, \tau y_i \mathbf{w}_2^{*T} \delta_i\right\} + max\left\{-y_i \mathbf{w}_1^{*T} \delta_i, \tau y_i \mathbf{w}_1^{*T} \delta_i\right\} \right]$$

$$\le \frac{C}{N} \sum_{i=1}^{N} \left\|\mathbf{w}_2^*\right\| \|\delta_i\| + \left\|\mathbf{w}_1^*\right\| \|\delta_i\|$$

$$\le \frac{C}{N} \sum_{i=1}^{N} 2\sqrt{C\varepsilon}$$

$$\le 2C\sqrt{C\varepsilon}$$

The scalable threshold ε is a very small normal number. It can be seen from Theorems 1–4 that compared with the pinball-SVM SVM trained with all training points, SKCHO-SVM proposed in this paper only uses the scalable kernel convex hull to train the classifier, but the optimal classification results of two methods are very close.

5 Experiments

In the following, we evaluated the performance of the proposed SKCHO-SVM method on four large-scale network traffic datasets and compare with four classification methods. Our experiments are organized as follows. Datasets and experiment settings are introduced in Sect. 5.1. The experimental analysis on SKCHO-SVM is presented in Sect. 5.2. In Sect. 5.3, the comparison experiments are displayed in terms of classification performance and running time.

In order to evaluate the robustness of SKCHO-SVM, two different noise strategies are designed in the experiments. First, following [22, 23], we add the noise points of 5% data

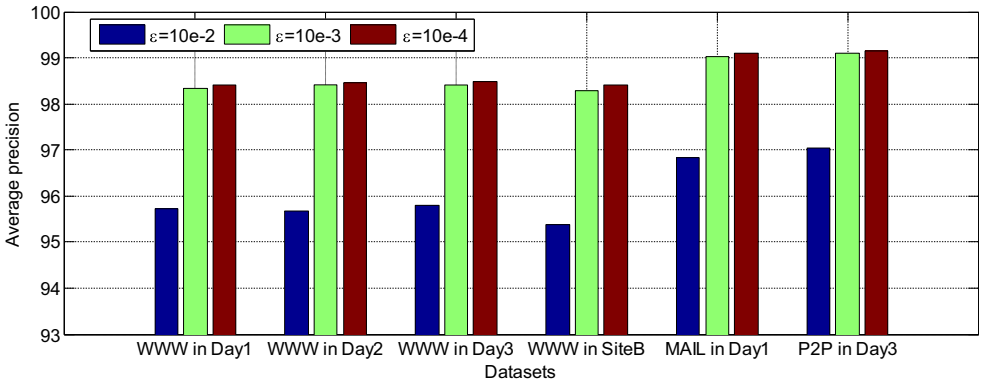

Fig. 1 Average precision of SKCHO-SVM with different values of parameter ε in noise mode 1

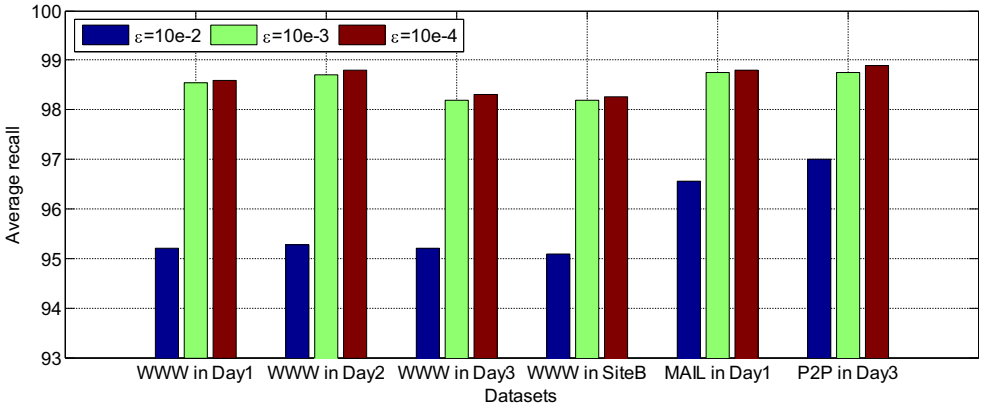

Fig. 2 Average recall of SKCHO-SVM with different values of parameter ε in noise mode 1

at the boundary and inside of the data. The added noises are 5% Gaussian white noise with the mean value of 0 and the variance of 5% of the data. The data boundary in the kernel space is obtained by SVDD method. Second, 2% data points are randomly selected to add Gaussian white noise with the mean value 0 and the variance 10% of the data. The intensity of noise mode 2 is greater than that of noise mode 1, and the number of noise points in noise mode 1 is greater than that in noise mode 2.

The datasets for online learning methods consist of three parts: the initial training dataset, the online learning dataset, and testing dataset. Following [27], we divide the data into ten equal and independent subsets, where five subsets for initial training, two subsets for online learning, and the rest three subsets for testing. The sets for online learning are equally divided into five parts, and all methods are updated with one part at a time.

5.1 Datasets and experiment settings

In our experiment, we use the classic network traffic data Moore datasets [29], which consists of several separate datasets each from a different period of the 24-h day. Each dataset is represented by tens of thousands of traffic flows, which are derived from header information. Four datasets named as Day1, Day2, Day3, and SiteB are selected for experimental comparisons. The traffic flows from Day1, Day2, and Day3 are selected on three weekdays in 2003, 2004, and 2006 from site A, and the ones form SiteB are selected on a weekday in 2007. The original data points in these four datasets are generated 248 features. Twelve features are selected by the correlation-based filtering mechanism following [30]. Table 1 shows the number of traffic flows of each class in each dataset.

From Table 1, we can easily see that the class imbalance exists in real word network traffic. The number of traffic flows from "WWW" accounts for the largest proportion of all flows. The number of traffic flows from "CHAT", "GAMES" and "GRID" are minimal. In this work, we delete the traffic flows of "GAMES" and "GRID" because they are very few. Network traffic classifications are generally evaluated from real-time, accuracy, and robustness. Real-time reflects the ability to classify online network traffic quickly. Accuracy and robustness reflect the ability to correctly classify network traffic and noise tolerance. In this study, we adopt training time, precision and recall as the performance indexes.

Since SVM is a classic binary classifier, we use the one-against-all classification strategy to perform multiclass classification problems. We train one SVM for each traffic category to separate the data points of this category from points of the other categories. The Gaussian kernel is used for all SVM classifiers. All parameters are selected by a five-fold cross-validation strategy on the initial training dataset. The regularization parameter and the kernel width σ in Gaussian kernel are taken in the set $\{10^{-3}, \ldots, 10^{3}\}and\{10^{-2}, \ldots, 10^{2}\}$, respectively. Scalable parameterεin pinball-SVM is selected in the set $\{10^{-4}, 10^{-3}, 10^{-2}\}$. The parameter τin pinball loss function is set 0.05. Based on extensive experiments, parameters β, θ,

Table 4 Average running time of offline learning stage of SKCHO-SVM and comparison methods on noise mode 1

	Pinball-SVM	LASVM	Online-FastKDE	OCVM	SKCHO-SVM
Day1	1794.50	254.01	9.98	23.18	13.87
Day2	986.27	111.77	5.84	13.01	7.22
Day3	1380.38	185.39	7.91	17.03	10.35
SiteB	1271.56	163.04	7.20	15.12	9.94

Table 5 Average performance for testing data points of all methods on Day1 in noise mode 1

Traffic classes	Pinball-SVM		LASVM		Online-FastKDE		OCVM		SKCHO-SVM	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)
WWW	98.21	98.27	90.53	90.43	86.86	86.99	97.30	97.11	*98.34*	*98.55*
MAIL	98.54	98.41	90.24	90.08	87.34	88.02	97.87	97.28	*99.03*	*99.12*
BULK	98.12	97.74	89.24	90.23	87.03	87.83	97.21	97.25	*98.47*	*98.70*
ATTACK	94.76	84.43	90.03	76.92	89.28	87.81	94.77	94.35	*95.02*	*95.14*
P2P	*96.49*	90.05	91.39	80.02	90.34	91.03	94.31	94.28	95.88	*95.64*
DATABASE	99.17	92.36	92.21	75.23	90.26	91.18	97.87	97.26	*99.21*	*99.09*
MULTIMEDIA	96.74	85.32	91.93	73.21	90.01	90.86	95.54	95.21	*96.98*	*96.84*
SERVICES	99.41	88.63	92.05	76.01	91.34	86.47	98.37	98.42	*99.43*	*99.26*

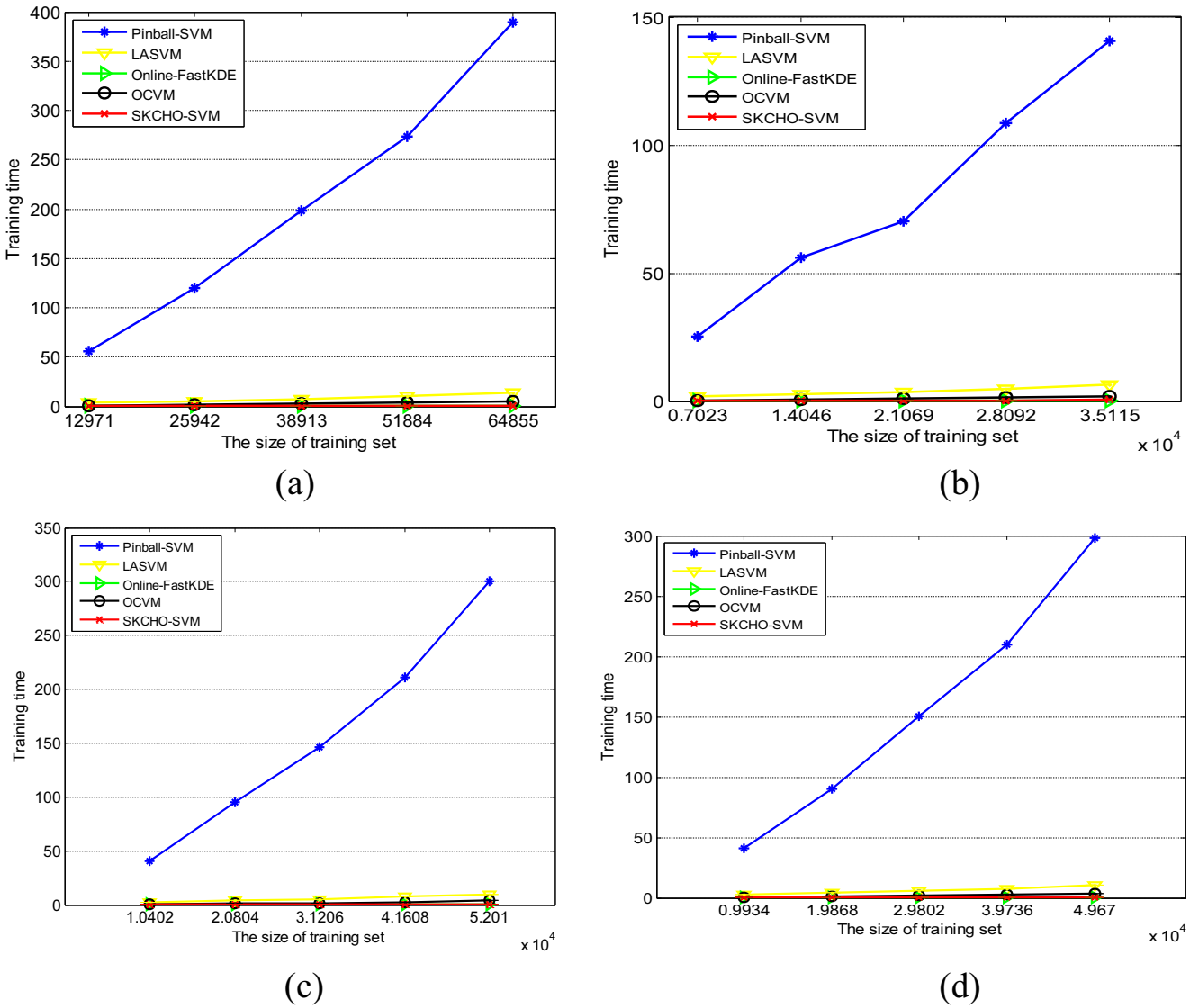

Fig. 3 Average running time of updating learning stage of SKCHO-SVM and four comparison methods on noise mode 1. **a** Day1. **b** Day2. **c** Day3. **d** SiteB

Table 6 Average performance (%) for testing data points of all methods on Day2 in noise mode 1

Traffic classes	Pinball-SVM		LASVM		Online-FastKDE		OCVM		SKCHO-SVM	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)
WWW	99.21	99.17	90.18	90.04	86.86	86.45	99.01	98.85	*99.23*	*99.27*
MAIL	*99.45*	*99.36*	89.66	89.83	87.34	86.63	99.12	98.76	*99.45*	99.14
BULK	99.17	*99.31*	88.31	88.46	87.03	86.77	99.06	98.53	*99.61*	99.20
ATTACK	93.74	86.93	89.05	80.99	89.28	89.01	93.47	93.1	*94.02*	*94.13*
P2P	97.39	85.54	90.42	78.01	90.34	90.06	97.78	97.27	*98.37*	*98.28*
DATABASE	99.02	84.36	92.01	79.55	90.24	90.02	98.80	98.39	*99.35*	*99.19*
SERVICES	99.06	85.06	92.28	82.28	91.34	90.65	98.55	98.02	*99.29*	*99.16*

and $\widetilde{\alpha}$ in SKCHO-SVM are set 0.1, 0.5, and 20, respectively. All parameters in comparison methods are obtained in their default settings. Our experiments are implemented in MATLAB using a computer with 2.6 GHz dual-core CPU, 8 GB RAM, and Windows operation system.

5.2 SKCHO-SVM on four network traffic datasets

An effective online learning strategy is to preserve the representative points and discard the un-representative ones in the newly arrived data. Which are representative and how much is enough to be two critical issues for online learning. In this subsection, the performance on SKCHO-SVM is presented as follows. We first discuss the percentage of scalable kernel convex hull in its corresponding class, and then discuss its running time and classification performance.

5.2.1 Average percentage of scalable kernel convex hull in its corresponding class and its running time

As discussed in Sect. 3, the number of scalable kernel convex hull vertices are related to the Gaussian kernel parameter σ and scalable parameter ε. We discuss the number of scalable kernel convex hull and running time with different σ and ε on six classification cases: WWW in Day1, WWW in Day2, WWW in Day3, WWW in SiteB, MAIL in Day1, and P2P in Day3. The experimental results in the cases of noise mode 1 are shown in Tables 2 and 3, respectively. We can see that:

1. The number of scalable kernel convex hull vertices increases with the increase of Gauss kernel parameter. This is because that the value of σ is related to the feature space. When its value is small, the distance between the points in the kernel space is small and the distribution is concentrated, so fewer convex hull vertices are obtained; conversely, when its value is large, the distance between the points is large and the distribution is dispersed, so more convex hull vertices are obtained.
2. When the scalable threshold is small, fewer points are satisfied Eq. (14), so more convex hull vertices are obtained, and the running time is longer. On the contrary, when the scalable threshold is large, fewer convex hull vertices are obtained and the running time is short.

5.2.2 Classification performance with different ε

In terms of Tables 2 and 3, the average percentage of scalable kernel convex hull in its corresponding class is

Table 7 Average performance (%) for testing data points of all methods on Day3 in noise mode 1

Traffic classes	Pinball-SVM		LASVM		Online-FastKDE		OCVM		SKCHO-SVM	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)
WWW	99.31	99.45	90.30	89.29	87.82	87.34	99.07	98.85	*99.33*	*99.46*
MAIL	99.22	91.03	90.25	82.14	85.47	85.20	98.89	98.78	*99.25*	*99.03*
BULK	99.10	90.01	89.03	81.71	85.61	85.31	98.72	98.56	*99.12*	*98.86*
P2P	99.16	*98.31*	89.99	87.52	86.19	86.86	98.57	98.51	*99.20*	*98.31*
DATABASE	99.29	91.05	89.37	82.06	87.24	86.72	99.55	98.94	*99.34*	*99.17*
INTERACTVE	*100*	90.38	92.75	79.05	88.13	87.66	100	98.78	*100*	*99.01*

Table 8 Average performance (%) for testing data points of all methods on SiteB in noise mode 1

Traffic classes	Pinball-SVM		LASVM		Online-FastKDE		OCVM		SKCHO-SVM	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)
WWW	98.21	98.12	90.44	88.44	89.65	87.69	98.12	98.97	*98.76*	*99.21*
MAIL	98.04	97.35	91.76	88.87	90.74	84.4	97.8	97.93	*98.62*	*98.43*
BULK	96.58	82.07	89.96	89.44	87.43	83.66	96.56	97.05	*97.39*	*97.64*
ATTACK	97.36	92.64	89.12	90.06	87.32	88.23	97.43	97.54	*98.17*	*98.05*
CHAT	95.40	80.35	88.05	88.05	88.7	85.06	94.99	88.43	*95.43*	*90.65*
P2P	98.12	98.02	90.3	90.42	88.32	82.16	97.04	97.02	*98.18*	*98.90*
VOIP	89.11	90.43	88.63	89.07	85.98	89.89	88.64	96.28	*89.32*	*97.07*

sensitive to scalable parameter ε. As we known, the number of and distribution of training data are directly related to the performance of SVM. In order to discuss the relationship of classification and scalable threshold ε, we experimentally study the average testing performance of SKCHO-SVM with different ε on six classification tasks: WWW in Day1, WWW in Day2, WWW in Day3, WWW in SiteB, MAIL in Day1, and P2P in Day3. Figures 1 and 2 show their precision and recall in noise mode 1, respectively. From these two figures, we can see that classification performance is closely related to scalable parameter ε. The smaller ε tends to obtain more scalable kernel convex hull vertices. The more vertex of convex hull, the longer running time of convex hull selection, but it will get better precision and recall rate. This is because the more convex hull vectors, the better the representation of contour distribution of data in the feature space. Therefore, in practical network traffic classification applications, we need to balance the running time and the number of convex hull vertices. In the following experiments, we fix $\varepsilon = 10^{-3}$.

5.3 Comparison experiments

In this section, we compare SKCHO-SVM with pinball-SVM (as baseline classifiers), and three online SVMs (including online-FastKDE [31], LASVM [32], and OCVM [27]) in terms of training time of offline learning stage, updating time of updating learning stage, and classification performance for testing stage. Pinball-SVM learns the classifier model in batch mode and updates the classifiers with old data points and newly arrived data points. Online-FastKDE is the online learning version of FastKDE method that updates the classifier based on the simple selection strategy. Online-FastKDE uses the simple point selection strategy to select the number of 1% training sets.

Average running time of offline learning stage of all comparison methods on noise mode 1 is presented in Table 4. We can see that the running time of offline learning stage of SKCHO-SVM is obviously less than other methods, except for Online-FastKDE. Since Online-FastKDE trains the classifier using the simple selection strategy, and only selects a small amount of data points for training. Pinball-SVM has

Table 9 Average performance (%) for testing data points of all methods on Day1 in noise mode 2

Traffic classes	Pinball-SVM		LASVM		Online-FastKDE		OCVM		SKCHO-SVM	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)
WWW	97.43	97.91	90.41	89.81	86.72	86.82	97.06	96.77	*98.10*	*98.51*
MAIL	98.02	98.35	90.19	89.93	87.28	87.84	97.64	96.58	*98.82*	*98.95*
BULK	97.63	97.34	88.81	89.51	86.95	87.24	97.00	96.70	*98.42*	*98.66*
ATTACK	94.13	83.89	89.84	76.26	88.87	87.67	94.51	93.83	*94.86*	*94.89*
P2P	*95.79*	89.34	90.95	79.31	90.07	90.37	94.28	94.20	95.60	*95.51*
DATABASE	98.91	92.10	91.55	74.58	90.18	90.85	97.56	96.88	*98.92*	*98.85*
MULTIMEDIA	96.29	84.83	91.84	73.17	89.61	90.35	95.48	94.85	*96.92*	*96.63*
SERVICES	98.98	88.12	91.78	75.37	91.14	85.99	98.08	97.74	*99.36*	*98.97*

Table 10 Average performance (%) for testing data points of all methods on Day2 in noise mode 2

Traffic classes	Pinball-SVM		LASVM		Online-FastKDE		OCVM		SKCHO-SVM	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)
WWW	98.86	98.45	89.72	89.51	86.83	86.21	98.33	98.25	*98.93*	*99.09*
MAIL	98.67	98.70	89.59	89.37	86.91	86.17	98.86	98.55	*99.29*	*99.14*
BULK	98.98	98.97	87.95	87.71	86.70	86.10	98.87	97.78	*99.35*	*99.07*
ATTACK	93.52	86.58	88.62	80.58	88.85	88.51	92.69	93.05	*93.99*	*93.91*
P2P	97.39	85.28	89.84	77.71	89.60	89.59	97.63	97.25	*98.30*	*98.18*
DATABASE	98.36	83.90	91.29	79.06	89.62	89.22	98.44	97.62	*99.08*	*98.90*
SERVICES	98.77	84.93	91.77	81.95	90.54	90.34	98.35	97.83	*99.02*	*99.10*

the longest running time of offline learning stage for all datasets. Since Pinball-SVM trains the classifier using the whole training set; thus, its number of training set is the largest among all comparison methods. The initial training of LASVM is based on SMO strategy called REPROCESS. Its training time of offline learning stage is larger than SKCHO-SVM. OCVM selects the convex hull as training data to train the classifier. The computation of OCVM is relatively complicated compared with SKCHO-SVM. Thus, the training time of OCVM for offline learning stage is also larger than that of SKCHO-SVM.

Average running time of updating learning stage of all comparison methods on noise mode 1 is presented in Fig. 3. We can see from Fig. 3 that the running time of SKCHO-SVM for offline learning stage is much shorter than those of pinball-SVM, LASVM, and OCVM. Meanwhile, the training time of SKCHO-SVM on four network traffic datasets is comparable with that of online-FastKDE. The reason is that SKCHO-SVM removes most of the redundant data points in the offline learning stage; thus, the classifier built on the remaining data points is much faster than pinball-SVM. Using the definition of scalable kernel convex hull in Eq. (12), the running time of extracting convex hull vertices is faster than LASVM and OCVM. In addition, we can see that with the increase of network traffic flows, the entire training time of SKCHO-SVM does not increase dramatically. Pinball-SVM uses batch mode to update the classifier, and its running time of online learning stage is much larger than other methods, with the increase of the newly arrived points.

The average classification performance (in terms of precision and recall) of all comparison methods on four network traffic datasets in noise mode 1 are shown in Tables 5, 6, 7, and 8. We can see that SKCHO-SVM obtains the best performance in terms of precision and recall. The reason is that (1) SKCHO-SVM deletes the noise points around the boundary of the data at the first period of offline learning stage, then SKCHO-SVM can obtain relatively "clean" data. Meanwhile, SKCHO-SVM is insensitive to noise with the help of pinball loss function. (2) SKCHO-SVM computes the scalable convex hull vertices in the kernel space based on Eq. (12). The obtained convex hull vertices are used as training data points both in the offline learning stage and online learning stage. The training data points of SKCHO-SVM are dynamically adjusted, so that the structure information of data can be retained well. (3) SKCHO-SVM efficiently relieves the imbalance problem in network traffic classification tasks. SKCHO-SVM respectively selects convex hull and obtains a small amount of points in different classes. Thus, the sizes of training data are roughly equal in different classes.

Table 11 Average performance (%) for testing data points of all methods on Day3 in noise mode 2

Traffic classes	Pinball-SVM		LASVM		Online-FastKDE		OCVM		SKCHO-SVM	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)
WWW	98.83	99.24	90.21	88.73	87.63	87.10	99.02	98.18	*99.11*	*99.31*
MAIL	98.50	90.79	89.76	81.65	85.02	84.86	98.46	97.98	*99.03*	*98.79*
BULK	98.50	89.48	88.92	80.95	85.42	85.08	97.96	98.18	*98.86*	*98.83*
P2P	98.49	*98.20*	89.95	87.50	85.41	86.66	98.28	98.04	*99.06*	98.14
DATABASE	99.27	90.59	88.62	81.65	86.95	86.33	99.07	98.33	*99.29*	*99.13*
INTERACTIVE	99.23	89.84	92.47	78.71	87.63	87.40	99.64	98.13	*99.81*	*98.80*

Table 12 Average performance (%) for testing data points of all methods on SiteB in noise mode 2

Traffic classes	Pinball-SVM		LASVM		Online-FastKDE		OCVM		SKCHO-SVM	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)
WWW	97.98	97.96	89.99	88.04	89.00	87.06	97.38	98.77	*98.62*	*98.96*
MAIL	97.54	96.84	91.13	88.85	90.67	84.01	97.45	97.78	*98.55*	*98.42*
BULK	96.49	81.48	89.90	88.66	87.43	83.37	95.90	97.04	*97.30*	*97.51*
ATTACK	97.20	92.17	88.85	89.60	87.00	87.55	96.81	97.27	*97.93*	*97.93*
CHAT	95.11	80.22	87.53	87.42	88.44	84.62	94.97	87.70	*95.23*	*90.65*
P2P	97.63	98.00	89.54	90.28	87.76	81.52	96.77	96.44	*98.17*	*98.88*
VOIP	**89.24**	90.31	88.44	88.86	85.23	89.56	88.43	95.50	89.12	*96.78*

In our experiments, we use the whole data points as training data one class when its size is less than 500.

The simple selection strategy in online-FastKDE cannot distinguish the noise from the normal points, so the noise will become the training points in offline learning stage and online learning stage. LASVM use REPROCESS strategy to remove nonsupport vectors during the update process. The main idea of REPROCESS depends on Karush-Kuhn-Tucker (KKT) condition of SVMs, so that LASVM is sensitive to noises and outliers. In addition, LASVM may discard some support vectors of classifier in online learning stage; thus, some important discriminative information may be discarded. For WWW, MAIL, and BULK classes, pinball-SVM achieves satisfactory classification performance, since the data points in these classes are sufficient. However, for ATTACK, P2P, DATABASE, MULTIMEDIA, and SERVICES classes, pinball-SVM achieves high precision and low recall since class imbalance leads to class hyperplane skew.

In view of noise distribution, noise points can be divided into two categories: one is distributed around the boundary of data points, and the other is distributed within the data area. In the following, we evaluate the second category of noise distribution. The average performance (in terms of precision and recall) of all comparison methods on four network traffic datasets in noise mode 2 are shown in Tables 9, 10, 11, and 12. The intensity of noise mode 2 is greater than that of noise mode 1; we can see that the classification performance of online-FastKDE and LASVM decrease with increasing noise intensity, both of which are sensitive to noise. The proposed SKCHO-SVM and pinball-SVM achieve the best precision and recall. The performance gap between them is lower than 0.1%. Under high intensity of noise, OCVM shows lower precision and recall than SKCHO-SVM and pinball-SVM. It is verified that pinball loss function is a robust loss function and insensitive to feature noise. Since OCVM can only delete noise points around the boundary of the data, and its precision and recall are lower than SKCHO-SVM when the noise points are distributed within the data area. The experimental results also show that using the scalable kernel convex hull, SKCHO-SVM is faster and more efficient than four comparison methods. Therefore, SKCHO-SVM is an effective tool for online network traffic classification tasks.

6 Conclusion

In this study, to solve large scale noise network traffic classification, we propose a novel online learning method called SKCHO-SVM, which takes advantages of both scalable kernel convex hull and pinball loss function. In the offline learning stage and online updating stage, SKCHO-SVM distinguishes noise points and uses scalable kernel convex hull to dynamically select a small amount of data points as training data. The selected scalable convex hull vertices can represent the profile of network traffic data in the kernel space. Thus, SKCHO-SVM efficiently builds an online network traffic classifier, which achieves the high classification performance and low computation complexity. However, we only study the feature noise problem in this paper. How to apply SKCHO-SVM to network traffic classification tasks with label noise is an interesting work in near future. In addition, the extraction attribution of network traffic flow is only twelve dimensions. We will further experimentally study the efficiency of SKCHO-SVM for high dimensional network traffic classification tasks.

Acknowledgments This work was supported in part by the National Natural Science Foundation of China under Grants 61976028 and 61806026 and by the Natural Science Foundation of Jiangsu Province under Grant BK 20180956.

References

1. China Internet Network Information Center (2019) Statistical Report on Internet Development in China. http://cnnic.com.cn/IDR/ReportDownloads/.[Online; Accessed 2-28-2019]

2. Nguyen T, Armitage G (2009) A survey of techniques for internet traffic classification using machine learning. IEEE Commun Surv Tutorials 10(4):56–76
3. Moore AW, Papagiannaki K (2005) Toward the accurate identification of network applications. In: Dovrolis C (ed) Passive and active network measurement. PAM 2005. Lecture Notes in Computer Science, vol 3431. Springer, Berlin, Heidelberg, pp 41–54. https://doi.org/10.1007/978-3-540-31966-5_4
4. Dainotti A, Pescape A, Claffy KC (2012) Issues and future directions in traffic classification. IEEE Netw 26(1):35–40
5. Bujlow T, Carela-Español V, Barlet-Ros P (2015) Independent comparison of popular DPI tools for traffic classification. Comput Netw 76(1):75–89
6. Surati S, Jinwala DC, Garg S (2017) A survey of simulators for P2P overlay networks with a case study of the P2P tree overlay using an event-driven simulator. Eng Sci Technol 20(2):705–720
7. Rezaei S, Liu X (2019) Deep learning for encrypted traffic classification: an overview. IEEE Commun Mag 57(5):76–81
8. Sun G, Chen T, Su Y, Li C (2018) Internet traffic classification based on incremental support vector machines. Mob Netw Applic 23(4):789–796
9. Gu XQ, Chung FL, Wang ST (2019) Extreme vector machine for fast training on large data. Int J Mach Learn Cybern 11:33–53. https://doi.org/10.1007/s13042-019-00936-3
10. Cao J, Fang Z, Qu G, Sun H, Zhang D (2017) An accurate traffic classification model based on support vector machines. Int J Netw Manag 27(1):1–15
11. Zhang J, Chen X, Xiang Y, Zhou W, Wu J (2015) Robust network traffic classification. IEEE/ACM Trans Networking 23(4):1257–1270
12. Divakaran DM, Su L, Liau YS, Thing VL (2015) SLIC: self-learning intelligent classifier for network traffic. Comput Netw 91(11):283–297
13. Ni TG, Gu XQ, Wang J, Zheng YH, Wang HY (2018) Scalable transfer support vector machine with group probabilities. Neurocomputing 273(1):570–582
14. Gu XQ, Chung FL, Wang ST (2018) Fast convex-hull vector machine for training on large-scale ncRNA data classification tasks. Knowl-Based Syst 151(6):149–164
15. Finamore A, Mellia M, Meo M, Rossi D (2010) Kiss: stochastic packet inspection classifier for udp traffic. IEEE/ACM Trans Netw 18(5):1505–1515
16. Gu C, Zhang S, Xue X (2011) Internet traffic classification based on fuzzy kernel K-means clustering. Int J Advancements in Comput Technol 3(3):199–209
17. Ertekin S, Bottou L, Giles CL (2011) Nonconvex online support vector machines. IEEE Trans Pattern Anal Mach Intell 33(2):368–381
18. Wang T, Chen J, Zhou Y, Snoussi H (2013) Online least squares one-class support vector machines-based abnormal visual event detection. Sensors 13(12):17130–17155
19. Wang D, Qiao H, Zhang B, Wang M (2013) Online support vector machine based on convex hull vertices selection. IEEE Trans Neural Netw Learn Syst 24(4):593–609
20. Wang J, Zhao P, Hoi SCH (2014) Cost-sensitive online classification. IEEE Trans Knowl Data Eng 26(10):2425–2438
21. Labovitz C, Johnson S, Oberheide J, Jahanian F, McPherson D (2010) Internet inter-domain traffic. In: Proceedings of the ACM SIGCOMM 2010 conference on applications, technologies, architectures, and protocols for computer communications. New Delhi, India, pp 75–86. https://doi.org/10.1145/1851182.1851194
22. Huang XL, Shi L, Suykens JAK (2014) Asymmetric least squares support vector machine classifiers. Comput Stat Data Anal 70(2):395–405
23. Huang XL, Shi L, Pelckmansb K, Suykens JAK (2014) Asymmetric ν-tube support vector regression. Comput Stat Data Anal 77(9):371–382
24. Huang XL, Shi L, Suykens JAK (2014) Support vector machine classifier with pinball loss. IEEE Trans Pattern Anal Mach Intell 36(5):984–997
25. Nandan M, Khargonekar PP, Talathi SS (2014) Fast SVM training using approximate extreme points. J Mach Learn Res 15(1):59–98
26. Almasi ON, Rouhani M (2016) Fast and de-noise support vector machine training method based on fuzzy clustering method for large real world datasets. Turk J Electr Eng Comput Sci 24(1):219–233
27. Ding S, Nie X, Qiao H, Zhang B (2013) A fast algorithm of convex hull vertices selection for online classification. IEEE Trans Neural Netw Learn Syst 29(4):792–806
28. David MJT (2004) Support vector data description. J Mach Learn Res 54(1):45–66
29. Network traffic data Moore datasets, https://www.cl.cam.ac.uk/research/srg/netos/projects/brasil/data/index.html [Online; Accessed 12-22-2018]
30. Li W, Canini M, Moore AW, Bolla R (2009) Efficient application identification and the temporal and spatial stability of classification schema. Comput Netw 53(6):790–809
31. Wang ST, Wang J, Chung F (2014) Kernel density estimation, kernel methods, and fast learning in large data sets. IEEE Trans Cybern 44(1):1–20
32. Bordes A, Ertekin S, Weston J, Bottou L (2005) Fast kernel classifiers with online and active learning. J Mach Learn Res 6(10):1579–1619